

## Практическое задание 3.

### Выпуклая негладкая, условная и структурная оптимизация.

**Начало выполнения задания:** 6 ноября 2016 г.

**Срок сдачи:** 30 ноября (среда), 23:59.

**Среда для выполнения задания:** Python 3.

## 1 Модель разреженной линейной регрессии

Рассматривается задача регрессии. Имеется обучающая выборка  $\mathcal{D} := ((x_i, y_i))_{i=1}^n$ , где  $x_i \in \mathbb{R}^d$  — вектор признаков  $i$ -го объекта, а  $y_i \in \mathbb{R}$  — его регрессионное значение. Задача заключается в прогнозировании регрессионного значения  $y_{\text{new}}$  для нового объекта, представленного своим вектором признаков  $x_{\text{new}}$ .

В линейной регрессии прогнозирование выполняется с помощью линейной функции:

$$y(x) := x^\top w,$$

где  $w \in \mathbb{R}^d$  — параметры модели, настраиваемые в процессе обучения.

Обучение модели осуществляется с помощью минимизации следующей  $L_1$ -регуляризованной функции потерь:

$$\phi(w) := \frac{1}{2n} \sum_{i=1}^n (x_i^\top w - y_i)^2 + \lambda \sum_{j=1}^d |w_j| =: \frac{1}{2n} \|Xw - y\|_2^2 + \lambda \|w\|_1 \rightarrow \min_{w \in \mathbb{R}^d}. \quad (1)$$

Здесь  $\lambda > 0$  — задаваемый пользователем коэффициент регуляризации. Использование  $L_1$ -регуляризации позволяет, во-первых, снизить вероятность переобучения модели, а, во-вторых, получить разреженное решение. В разреженном решении часть компонент оптимального вектора весов  $w^*$  равна нулю; (можно показать, что при  $\lambda \geq \|X^\top y\|_\infty$  все веса будут равны нулю). Нулевые веса соответствуют исключению соответствующих признаков из модели (признание их неинформативными).

### 1.1 Двойственная задача и критерий остановки

Перепишем задачу (1) в следующем эквивалентном виде:

$$\min_{w \in \mathbb{R}^d, z \in \mathbb{R}^n} \left\{ \frac{1}{2n} \|z\|_2^2 + \lambda \|w\|_1 : Xw - y = z \right\}. \quad (2)$$

Можно показать, что двойственной задачей к задаче (2) является

$$\max_{\mu \in \mathbb{R}^n} \left\{ -\frac{n}{2} \|\mu\|_2^2 - y^\top \mu : \|X^\top \mu\|_\infty \leq \lambda \right\}. \quad (3)$$

Таким образом, имея в распоряжении допустимую двойственную точку  $\mu \in \mathbb{R}^n$ , т. е. такую что  $\|X^\top \mu\|_\infty \leq \lambda$ , можно вычислить следующую оценку для невязки в задаче (1):

$$\phi(w) - \phi^* \leq \frac{1}{2n} \|Xw - y\|_2^2 + \lambda \|w\|_1 + \frac{n}{2} \|\mu\|_2^2 + y^\top \mu =: \eta(w, \mu). \quad (4)$$

Величина  $\eta(w, \mu)$  называется *зазором двойственности* и обращается в ноль в оптимальных решениях  $w^*$  и  $\mu^*$  задач (1) и (3). Заметим, что решения  $w^*$  и  $\mu^*$  связаны между собой следующим соотношением:  $Xw^* - y = n\mu^*$ . Поэтому для фиксированного  $w$  естественным выбором соответствующего  $\mu$  будет

$$\mu(w) := \min \left\{ 1, \frac{n\lambda}{\|X^\top(Xw - y)\|_\infty} \right\} \frac{1}{n}(Xw - y).$$

Такой выбор обеспечивает стремление зазора двойственности  $\eta(w, \mu(w))$  к нулю при  $w \rightarrow w^*$ , что позволяет использовать условие  $\eta(w, \mu(w)) < \epsilon$  в качестве критерия останова в любом итеративном методе решения задачи (1).

## 1.2 Сведение к гладкой условной задаче оптимизации

Введя вспомогательные переменные  $w^+, w^-$ , задачу (1) можно эквивалентно переписать в виде гладкой условной задачи

$$\min_{w^+ \in \mathbb{R}^d, w^- \in \mathbb{R}^d} \left\{ \frac{1}{2n} \|X(w^+ - w^-) - y\|_2^2 + \lambda \bar{\Gamma}^\top(w^+ + w^-) : w^+ \succeq 0 \wedge w^- \succeq 0 \right\}. \quad (5)$$

где  $\bar{\Gamma} := (1, \dots, 1)$ .

Задачу (5) можно решать с помощью метода барьеров.

## 2 Субградиентный метод

Субградиентный метод — это общий метод решения негладкой выпуклой задачи оптимизации

$$\min_{x \in \mathbb{R}^n} \phi(x),$$

где  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$  — выпуклая функция, возможно, недифференцируемая.

Итерация субградиентного метода заключается в шаге из текущей точки  $x_k$  в направлении (любого) анти-субградиента  $\phi'(x_k)$ . При этом, поскольку для негладких задач норма субградиента  $\|\phi'(x_k)\|_2$  не является информативной, субградиентный метод использует в качестве направления нормированный вектор  $\phi'(x_k)/\|\phi'(x_k)\|_2$ :

$$x_{k+1} = x_k - \alpha_k \frac{\phi'(x_k)}{\|\phi'(x_k)\|_2}.$$

Для сходимости метода необходимо, чтобы длины шагов  $\{\alpha_k\}_{k \geq 0}$  убывали к нулю, но не слишком быстро:

$$\alpha_k > 0, \quad \alpha_k \rightarrow 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty.$$

Обычно длины шагов выбирают по правилу  $\alpha_k = \alpha/\sqrt{k+1}$ , где  $\alpha > 0$  — некоторая константа.

Нужно отметить, что последовательность  $\{x_k\}$ , построенная субградиентным методом, может не быть релаксационной последовательностью для функции  $\phi$ , т. е. утверждение  $\phi(x_{k+1}) \leq \phi(x_k)$  может быть неверно. Поэтому в субградиентном методе в качестве результата работы после  $N$  итераций метода вместо точки  $x_N$  возвращается точка  $y_N := \operatorname{argmin}\{\phi(x) : x \in \{x_0, \dots, x_N\}\}$  (т. е. из всех пробных точек  $x_k$ , построенных методом, выбирается та, в которой значение функции оказалось наименьшим).<sup>1</sup>

<sup>1</sup>Заметим, что в практической реализации метода для вычисления результата  $y_N$  сами точки  $x_0, \dots, x_N$  хранить в памяти не нужно.

### 3 Проксимальный градиентный метод

Проксимальный градиентный метод используется для минимизации *композиционных функций*:

$$\phi(x) := f(x) + h(x),$$

где функция  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  непрерывно дифференцируемая выпуклая функция, а функция  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  выпуклая и простая, возможно, недифференцируемая.

Под *простотой* функции  $h$  подразумевается то, что для этой функции возможно эффективно вычислить проксимальное отображение

$$\text{prox}_h^\alpha(x) := \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} \left[ \alpha h(y) + \frac{1}{2} \|y - x\|_2^2 \right],$$

где  $x \in \mathbb{R}^n$ ,  $\alpha > 0$ . Например, для  $h(x) = \|x\|_1$  проксимальное отображение может быть вычислено аналитически независимо для каждой из координат  $i = 1, \dots, n$ :

$$[\text{prox}_{\|\cdot\|_1}^\alpha(x)]_i = \begin{cases} x_i + \alpha, & x_i < -\alpha, \\ 0, & |x_i| \leq \alpha, \\ x_i - \alpha, & x_i > \alpha. \end{cases}$$

Итерация проксимального градиентного метода имеет следующий вид:

$$x_{k+1} = \text{prox}_h^{\alpha_k}(x_k - \alpha_k \nabla f(x_k)), \quad (6)$$

где  $\nabla f(x_k)$  — градиент дифференцируемой функции  $f$  в точке  $x_k$ , а  $\alpha_k > 0$  — некоторым образом выбранная длина шага.

#### 3.1 Схема Нестерова для подбора длины шага

Итерация (6) имеет следующий геометрический смысл. Если градиент функции  $f$  удовлетворяет условию Липшица с константой  $L_f > 0$ , то для любых  $x, y \in \mathbb{R}^n$  и  $L \geq L_f$  справедлива оценка

$$\phi(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2} \|y - x\|_2^2 + h(y) =: m_L(y; x), \quad (7)$$

которая является точной в точке  $y = x$ . Выбрав  $x = x_k$ ,  $L = L_k := 1/\alpha_k$ , получаем, что итерация проксимального градиентного метода (6) в точности соответствует минимизации функции  $m_{L_k}(\cdot; x_k)$ :

$$x_{k+1} = \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} m_{L_k}(y; x_k).$$

Если  $L_k \geq L_f$ , то, согласно оценке (7), значение функции  $\phi$  в новой точке  $x_{k+1}$  уменьшается относительно значения в старой точке  $\phi(x_k)$  по крайней мере на величину  $m_{L_k}(x_{k+1}; x_k) - \phi(x_k)$ :

$$\phi(x_{k+1}) \leq m_{L_k}(x_{k+1}; x_k) \quad (\leq \phi(x_k)). \quad (8)$$

Оказывается, что именно это неравенство, а не условие  $L_k \geq L_f$  отвечает за сходимость проксимального градиентного метода. Поэтому для подбора константы  $L_k$  (или, эквивалентно, длины шага  $\alpha_k$ ) на текущей итерации  $k$  можно использовать следующую простую процедуру одномерного поиска: начать с некоторого значения  $L$  и увеличивать его в два раза, пока не выполнится неравенство (8). Заметим, что эта процедура определена корректно: как только значение  $L$  превысит  $L_f$ , неравенство (8) обязательно будет выполнено в силу липшицевости градиента функции  $f$ . О константе  $L_k$  удобно думать как о «локальной» константе Липшица градиента функции  $f$ .

Поскольку каждое пробное значение константы  $L$  требует полного вычисления функции  $\phi$ , то число итераций одномерного поиска необходимо, по возможности, сократить. Для этого имеет смысл инициализировать значение  $L_{k+1}$  с помощью уже найденного значения  $L_k$ . Естественным вариантом является инициализация  $L_{k+1} = L_k$ . Однако в этом случае константы  $L_k$  всегда будут только увеличиваться и никогда не уменьшаться; это плохо, поскольку при прочих равных лучше выбрать значение  $L$  как можно меньше, что будет соответствовать большему шагу и большему прогрессу в оптимизации (в некоторых областях пространства локальная константа Липшица может быть меньше, чем в области, содержащей начальную точку метода). Таким образом, чтобы константы Липшица на соседних итерациях были близки, но также со временем могли уменьшаться, в схеме Нестерова выполняется инициализация  $L_{k+1} = L_k/2$ .

Итоговый алгоритм проксимального градиентного метода с подбором длины шага по схеме Нестерова представлен в алгоритме 1. Здесь  $L_0 > 0$  — параметр метода (нижняя оценка на глобальную константу Липшица  $L_f$ ); его всегда можно выбрать равным единице ( $L_0 = 1$ ) без принципиального ущерба для сходимости метода.

Можно показать, что, несмотря на то, что на отдельных итерациях проксимального градиентного метода может выполняться много шагов одномерного поиска, среднее число вычислений функции  $\phi$  за итерацию равно двум (задание 4).

```

1  $L \leftarrow L_0$ ;
2 for  $k = 0, 1, 2, \dots$  do
3   repeat
4      $y \leftarrow \operatorname{argmin}_{y \in \mathbb{R}^n} m_L(y; x_k)$ ;
5     if  $\phi(y) > m_L(y; x_k)$  then  $L \leftarrow 2L$ ;
6   until  $\phi(y) \leq m_L(y; x_k)$ ;
7    $x_{k+1} \leftarrow y$ ;
8    $L \leftarrow \max\{L_0, L/2\}$ ;
9 end

```

**Алгоритм 1:** Проксимальный градиентный метод с подбором длины шага по схеме Нестерова

## 4 Формулировка задания

1. Выписать для задачи (5) вспомогательную функцию  $\phi_t(w^+, w^-)$ , минимизируемую в методе барьеров. Выписать систему линейных уравнений, задающую ньютоновское направление  $p_k = (p_k^+, p_k^-)$ , предложить свой способ решения этой системы и прокомментировать его достоинства и недостатки. Для текущей точки  $(w_k^+, w_k^-)$  и направления  $p_k$  определить максимальную допустимую длину шага  $\alpha$ . Какую начальную точку  $(w_0^+, w_0^-)$  можно выбрать для метода барьеров?
2. Реализовать метод барьеров для задачи (5). Для решения систем линейных уравнений можно использовать стандартные алгоритмы линейной алгебры из `numpy/scipy`. Для подбора длины шага использовать последовательное деление пополам до выполнения условия Армихо для функции  $\phi_t$  (не забыть выбирать начальное значение длины шага равным единице, если оно допустимо). В качестве критерия остановки использовать зазор двойственности (4).
3. Реализовать для задачи (1) субградиентный метод и проксимальный градиентный метод. В качестве критерия остановки в обоих методах использовать зазор двойственности (4). Какую начальную точку можно выбрать для субградиентного метода? Какую для проксимального?
4. Для проксимального метода построить график суммарного (кумулятивного) числа итераций одномерного поиска в зависимости от номера итерации. Действительно ли среднее число итераций одномерного поиска не превышает (примерно) двух?

5. Провести экспериментальное сравнение трех реализованных методов для различных значений числа переменных  $d$ ; рассмотреть, как минимум, случаи  $d \leq 100$  и  $d \geq 1000$ . Какие методы быстрее достигают низкой точности ( $\epsilon = 10^{-2}$ ), а какие высокой ( $\epsilon = 10^{-10}$ )?

Значение коэффициента регуляризации  $\lambda$  выбрать стандартным образом:  $\lambda = 1/n$ .

Для сравнения методов рисовать графики 1) числа итераций против гарантируемой точности по зазору двойственности (4) и 2) реального времени работы против зазора двойственности. Для зазора двойственности использовать логарифмическую шкалу.

**Рекомендация:** Реальные наборы данных для тестирования можно взять с сайта LIBSVM<sup>2</sup>. Любой набор данных с сайта LIBSVM представляет из себя текстовый файл в формате svmlight. Чтобы считать такой текстовый файл, можно использовать функцию `load_svmlight_file` из модуля `sklearn.datasets`. Эта функция всегда возвращает матрицу  $X$  типа `sp.sparse.csr_matrix` (разреженная матрица). В данном задании разреженность матрицы  $X$  никак использовать не нужно, поэтому сразу же после вызова функции `load_svmlight_file` следует привести  $X$  к типу `np.ndarray`. Это можно сделать с помощью команды `X = X.toarray()`.

6. Написать отчет в формате PDF с описанием всех проведенных исследований.

## 5 Оформление задания

Результатом выполнения задания являются 1) pdf-отчет о проведенных исследованиях со всеми необходимыми формулами и выводами, 2) файл `l1linreg.py`, содержащий исходные коды трех методов. Выполненное задание следует отправить письмом по адресу `bayesml@gmail.com` с заголовком

«[BMK MOMO16] Задание 3, Фамилия Имя».

Убедительная просьба присылать выполненное задание только один раз с окончательным вариантом.

Поскольку проверка реализованных алгоритмов будет осуществляться в полуавтоматическом режиме, все реализованные функции должны строго соответствовать приведенным ниже прототипам и корректно запускаться в Python 3 (а не Python 2!). Проверить наличие всех необходимых функций, а также их соответствие требуемым прототипам можно с помощью специального скрипта `check_submission_ass3.py`, выдаваемого вместе с текстом задания.

## 6 Прототипы функций

1. Метод барьеров для решения задачи (5):

Модуль:	<code>l1linreg</code>
Функция:	<code>barrier(X, y, reg_coef, w0_plus, w0_minus, tol=1e-5, tol_inner=1e-8, max_iter=100, max_iter_inner=20, t0=1, gamma=10, c1=1e-4, disp=False, trace=False)</code>
Параметры:	$X$ : <code>np.ndarray</code> Матрица признаков размеров $n \times d$ . $y$ : <code>np.ndarray</code> Регрессионные значения, вектор размера $n$ . <code>reg_coef</code> : <code>float</code> Коэффициент регуляризации $\lambda > 0$ .

<sup>2</sup><http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>.

	<p><b>w0_plus:</b> np.ndarray</p> <p>Начальная точка <math>w_0^+</math>, вектор размера <math>d</math>.</p> <p><b>w0_minus:</b> np.ndarray</p> <p>Начальная точка <math>w_0^-</math>, вектор размера <math>d</math>.</p> <p><b>tol:</b> float, опционально</p> <p>Точность оптимизации по функции: <math>\phi(\hat{w}) - \phi^* \leq \epsilon</math>.</p> <p><b>tol_inner:</b> float, опционально</p> <p>Точность оптимизации функции <math>\phi_t</math> на внутренних итерациях метода Ньютона:  <math>\ \nabla \phi_t(w_k^+, w_k^-)\ _\infty &lt; \epsilon^{\text{inner}}</math>.</p> <p><b>max_iter:</b> int, опционально</p> <p>Максимальное число (внешних) итераций метода.</p> <p><b>max_iter_inner:</b> int, опционально</p> <p>Максимальное число внутренних итераций метода Ньютона.</p> <p><b>t0:</b> float, опционально</p> <p>Начальное значение параметра центрирования <math>t</math>.</p> <p><b>gamma:</b> float, опционально</p> <p>Коэффициент увеличения параметра <math>t</math> на внешних итерациях: <math>t_{k+1} = \gamma t_k</math>.</p> <p><b>c1:</b> float, опционально</p> <p>Значение константы <math>c_1</math> в условии Армихо.</p> <p><b>disp:</b> bool, опционально</p> <p>Отображать прогресс метода по итерациям (номер итерации, пройденное время, значение функции, зазор двойственности и пр.) или нет.</p> <p><b>trace:</b> bool, опционально</p> <p>Сохранять траекторию метода для возврата истории или нет.</p>
Возврат:	<p><b>w_hat:</b> np.ndarray</p> <p>Найденная точка <math>\hat{w}</math>, вектор размера <math>d</math>.</p> <p><b>status:</b> int</p> <p>Статус выхода, число:</p> <ul style="list-style-type: none"> <li>0: решение найдено с заданной точностью: <math>\phi(\hat{w}) - \phi^* \leq \epsilon</math>;</li> <li>1: достигнуто максимальное число итераций.</li> </ul> <p><b>hist:</b> dict, возвращается только если <b>trace=True</b></p> <p>История процесса оптимизации по итерациям (новая запись добавляется на каждой внутренней итерации метода). Словарь со следующими полями:</p> <p><b>elaps_t:</b> np.ndarray</p> <p>Время, пройденное с начала оптимизации.</p> <p><b>phi:</b> np.ndarray</p> <p>Текущее значение функции <math>\phi(w_k)</math>.</p> <p><b>dual_gap:</b> np.ndarray</p> <p>Текущий зазор двойственности <math>\eta(w_k, \mu(w_k))</math>, заданный в (4).</p>

## 2. Субградиентный метод для решения задачи (1):

Функция:	<code>subgrad(X, y, reg_coef, w0, tol=1e-2, max_iter=1000, alpha=1, disp=False, trace=False)</code>
Параметры:	<p><code>w0: np.ndarray</code> Начальная точка <math>w_0</math>, вектор размера <math>d</math>.</p> <p><code>alpha: float</code>, опционально Константа <math>\alpha</math> для выбора длины шага: <math>\alpha_k = \alpha / \sqrt{k+1}</math>.</p> <p>Остальные параметры такие же, как и для функции <code>barrier</code>.</p>

### 3. Проксимальный метод для решения задачи (1):

Функция:	<code>prox_grad(X, y, reg_coef, w0, tol=1e-5, max_iter=1000, L0=1, disp=False, trace=False)</code>
Параметры:	<p><code>w0: np.ndarray</code> Начальная точка <math>w_0</math>, вектор размера <math>d</math>.</p> <p><code>L0: float</code>, опционально Константа <math>L_0</math> в схеме Нестерова для подбора длины шага.</p> <p>Остальные параметры такие же, как и для функции <code>barrier</code>.</p>

**Внимание:** В этом методе история `hist` должна содержать дополнительное четвертое поле `ls_iters`, содержащее суммарное (кумулятивное) число итераций одномерного поиска с самого начала работы проксимального метода.